PLDs (combinatorial circuits): ROM, PLA, PAL, CPLD, and FPGA

Store *permanent* binary information (nonvolatile). Can be read only (cannot be altered). Information is specified by designer and *physically inserted* (embedded) into the PLD.

Programmable connections are formed by *fuses*, *masks*, or *antifuses* depending on the technology. Irreversible programming.
$k$ inputs (address) $\Rightarrow$ $2^k \times n$ ROM $\Rightarrow$ $n$ outputs (data)

- $k \times 2^k$ decoder to decode input address

- $n$ OR gates with $2^k$ input each

- Decoder output is connected to all $n$ OR gates through fuses

- ROM $\rightarrow$ $2^k \times n$ programmable connections
Programming a ROM

Example of $4 \times 2$ ROM

<table>
<thead>
<tr>
<th>Truth table</th>
<th>NonProgrammed ROM</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Address</strong></td>
<td><strong>Content</strong></td>
</tr>
<tr>
<td>$I_1$</td>
<td>$I_0$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Programmed ROM

Compact ROM

Truth table $\rightarrow$ address and content of ROM

**Programming** $\rightarrow$ stores truth table in ROM

- 0 = Open connection = Fuse blown
- 1 = Closed connection = Fuse intact
Any set of functions $f_1(x_k, \ldots, x_1), \ldots, f_n(x_k, \ldots, x_1)$ can be realized with a $2^k \times n$ ROM.

**Example:** Implement $f_1(x_2, x_1) = \sum m(0, 3)$, $f_2(x_2, x_1) = x_2 + x_1$, and $f_3(x_2, x_1) = \prod M(1)$ with a $4 \times 3$ ROM.

<table>
<thead>
<tr>
<th>Address</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_2$</td>
<td>$x_1$</td>
</tr>
<tr>
<td>0 0</td>
<td></td>
</tr>
<tr>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td></td>
</tr>
</tbody>
</table>

4 \times 3 ROM storing $f_1, f_2, f_3$
Programmable Logic Array

Behave like a ROM but has different structure

- Uses ANDs array instead of decoder to produce product terms of inputs

- Has programmable connections before ANDs, between ANDs and ORs, after ORs. That is $2nk + km + m$ fuses

- More flexible than ROM but more difficult to program

- Logic expressions for content information to be stored in PLA must be obtained first, then minimized, and finally programmed into the PLA using a PLA program table

- PLA program table specifies product terms and sum terms of information that will be stored in PLA
Programming a PLA

PLA Program Table

<table>
<thead>
<tr>
<th>Term</th>
<th>Term#</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AB$</td>
<td>1</td>
<td>1 0 _</td>
<td>1 _</td>
</tr>
<tr>
<td>$AC$</td>
<td>2</td>
<td>1 _ 1</td>
<td>1 1</td>
</tr>
<tr>
<td>$BC$</td>
<td>3</td>
<td>_ 1 1</td>
<td>_ 1</td>
</tr>
<tr>
<td>$\overline{ABC}$</td>
<td>4</td>
<td>0 1 0</td>
<td>1 _</td>
</tr>
</tbody>
</table>

Corresponding PLA Implementation

X Fuse intact
+ Fuse blown
Any set of functions $f_1(x_1, \ldots, x_n), \ldots, f_m(x_1, \ldots, x_n)$ can be realized with a PLA.

**Example** Implement $f_1(a, b, c) = \sum m(3, 5, 6, 7)$ and $f_2(a, b, c) = \sum m(0, 2, 4)$ with a PLA.

**First** Simplify $f_1, \overline{f_1}, f_2, \overline{f_2}$, that is

- $f_1(a, b, c) = ab + ac + bc$ \quad $f_1, f_2 \rightarrow 5$ terms
- $\overline{f_1}(a, b, c) = \overline{ab} + \overline{ac} + \overline{bc}$ \quad $f_1, \overline{f_2} \rightarrow 4$ terms
- $f_2(a, b, c) = \overline{ac} + \overline{bc}$ \quad $\overline{f_1}, f_2 \rightarrow 3$ terms
- $\overline{f_2}(a, b, c) = ab + c$ \quad $\overline{f_1}, \overline{f_2} \rightarrow 5$ terms

**Second** Select combination of functions that has less terms, that is

- $f_1 = \overline{f_1} = \overline{ab} + \overline{ac} + \overline{bc}$
- $f_2(a, b, c) = \overline{ac} + \overline{bc}$

**Third** Construct a PLA program table from selected functions

<table>
<thead>
<tr>
<th>Term</th>
<th>Term#</th>
<th>Inputs $a$ $b$ $c$</th>
<th>Outputs $f_1$ $f_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\overline{ab}$</td>
<td>1</td>
<td>0 0 _</td>
<td>1  _</td>
</tr>
<tr>
<td>$\overline{ac}$</td>
<td>2</td>
<td>0 _ 0</td>
<td>1  1</td>
</tr>
<tr>
<td>$\overline{bc}$</td>
<td>3</td>
<td>_ 0 0</td>
<td>1  1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>C  T</td>
</tr>
</tbody>
</table>
**Function Synthesis with PLA**
(continued)

**Third** Construct a PLA program table from selected functions

<table>
<thead>
<tr>
<th>Term</th>
<th>Term#</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\overline{ab}$</td>
<td>1</td>
<td>0 0 _</td>
<td>1 _</td>
</tr>
<tr>
<td>$\overline{ac}$</td>
<td>2</td>
<td>0 _ 0</td>
<td>1 1</td>
</tr>
<tr>
<td>$\overline{bc}$</td>
<td>3</td>
<td>_ 0 0</td>
<td>1 1</td>
</tr>
</tbody>
</table>

**Fourth** Construct PLA circuit from PLA program table

![PLA Circuit Diagram](image-url)
Programmable Array Logic

Similar to PLA

- Only the connection inputs to ANDs are programmable
- Easier to program than but not as flexible as PLA
- There are feedback connections
- Logic expressions for content information to be stored in PAL must be obtained first, then minimized, and finally programmed into the PAL using a PAL program table
- PAL program table specifies only product terms of information that will be stored in PAL
Programming a PAL

AND gates inputs

Product term

1
2
3
4
5
6
7
8
9
10
11
12

A
B
C
D

W

X

Y

Z

All fuses intact
(always = 0)

X Fuse intact

+ Fuse blown
Essential element of the Central Processing Unit

Arithmetic and logic functions on binary words

- $n$-bit data inputs $A$ and $B$
- $n$-bit data output $G = f(A, B)$
- Selection inputs $S_0, S_1$ select a function $f$
- Selection input $S_2$ select an operating mode (arithmetic or logic)
ALU  
(continued)

**Logic Circuit**

(a) Logic Diagram

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_0$</th>
<th>Output</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$G = A \land B$</td>
<td>AND</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$G = A \lor B$</td>
<td>OR</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$G = A \oplus B$</td>
<td>XOR</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$G = \overline{A}$</td>
<td>NOT</td>
</tr>
</tbody>
</table>

(b) Function Table

**Arithmetic Circuit**

Input logic

$n$-bit parallel adder

$G = X + Y + C_{in}$

$C_{out}$
ALU
(continued)
ALU
(continued)

Arithmetic and Logic Circuit

One stage of arithmetic circuit

One stage of logic circuit

2-to-1 MUX

$C_1$

$A_i$

$B_i$

$S_0$

$S_1$

$S_2$

$C_{i+1}$

$G_i$